Terms used **function inheritance** Found **71,194** of **147,060**

Sort results by relevance Save results to a Binder Try an Advanced Search

Display results expanded form Search Tips Try this search in The ACM Guide

☐ Open results in a new window

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown Relevance scale

1 Inheritance is not subtyping

William R. Cook, Walter Hill, Peter S. Canning

December 1989 **Proceedings of the 17th ACM SIGPLAN-SIGACT symposium on Principles of programming languages**

Full text available: pdf(1.24 MB) Additional Information: full citation, abstract, references, citings, index terms

In typed object-oriented languages the subtype relation is typically based on the inheritance hierarchy. This approach, however, leads either to insecure type-systems or to restrictions on inheritance that make it less flexible than untyped Smalltalk inheritance. We present a new typed model of inheritance that allows more of the flexibility of Smalltalk inheritance within a statically-typed system. Significant features of our analysis are the introduction of polymorphism into the typing of ...

2 DDD papers: The power of symmetry: unifying inheritance and generative programming

DeLesley Hutchins

October 2003 **Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**

Full text available: pdf(567.86 KB) Additional Information: full citation, abstract, references, index terms, review

I present the Ohmu language, a unified object model which allows a number of "advanced" techniques such as aspects, mixin layers, parametric polymorphism, and generative components to be implemented cleanly using two basic concepts: block structure and inheritance. I argue that conventional ways of defining classes and objects have created artificial distinctions which limit their expressiveness. The Ohmu model unifies functions, classes, instances, templates, and even aspects into a single cons ...

Keywords: aspect-oriented programming, aspects, code generation, covariant specialization, generative components, generative programming, generic types, join points, meta-programming, mixin layers, mixins, multiple inheritance, parametric polymorphism, partial evaluation, prototypes, transformation systems, virtual classes, virtual types

3 Compiling inheritance using partial evaluation

Siau Cheng Khoo, R. S. Sundaresh

May 1991 **ACM SIGPLAN N tices , Pr ceedings f the 1991 ACM SIGPLAN symposium**

n Partial evaluation and semantics-based program manipulati n, Volume 26 Issue 9
Full text available: pdf(819.28 KB) Additional Information: full citation, references, citings, index terms

4 LLO: an object-oriented deductive language with methods and method inheritance
Yanjun Lou, Z. Meral Ozsoyoglu
April 1991 **ACM SIGMOD Record , Proceedings of the 1991 ACM SIGMOD international conference on Management of data**, Volume 20 Issue 2
Full text available: pdf(1.22 MB) Additional Information: full citation, references, citings, index terms

5 CLOS: integrating object-oriented and functional programming
Richard P. Gabriel, Jon L. White, Daniel G. Bobrow
September 1991 **Communications of the ACM**, Volume 34 Issue 9
Full text available: pdf(2.53 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Lisp has a long history as a functional language,* where action is invoked by calling a procedure, and where procedural abstraction and encapsulation provide convenient modularity boundaries. A number of attempts have been made to graft object-oriented programming into this framework without losing the essential character of Lisp—to include the benefits of data abstraction, extensible type classification, incremental operator definition, and code reuse through an ...

Keywords: Common Lisp Object Systems

6 Enhancement for multiple-inheritance
James Hendler
June 1986 **ACM SIGPLAN Notices , Proceedings of the 1986 SIGPLAN workshop on Object-oriented programming**, Volume 21 Issue 10
Full text available: pdf(755.53 KB) Additional Information: full citation, references, citings, index terms

7 Forcing behavioral subtyping through specification inheritance
Krishna Kishore Dhara, Gary T. Leavens
May 1996 **Proceedings of the 18th international conference on Software engineering**
Full text available: pdf(1.09 MB) Publisher Site Additional Information: full citation, abstract, references, citings, index terms

A common change to object-oriented software is to add a new type of data that is a subtype of some existing type in the program. However, due to message passing, unchanged pearls of the program may now call operations of the new type. To avoid reverification of unchanged code, such operations should have specifications that are related to the specifications of the appropriate operations in their supertypes. This paper presents a specification technique that uses inheritance of specifications to ...

Keywords: C++, behavioral subtyping, code reverification, formal specification, inheritance, message passing, object-oriented programming, object-oriented software, specification inheritance

8

F-logic: a higher-order language for reasoning about objects, inheritance, and scheme
Michael Kifer, Georg Lausen
June 1989 **ACM SIGMOD Rec rd , Pr ceedings of the 1989 ACM SIGMOD international conference on Management f data**, Volume 18 Issue 2
Full text available: pdf(1.58 MB) Additional Information: full citation, abstract, references, citings, index terms

We propose a database logic which accounts in a clean declarative fashion for most of the "object-oriented" features such as object identity, complex objects, inheritance, methods, etc. Furthermore, database schema is part of the object language, which allows the user to browse schema and data using the same declarative formalism. The proposed logic has a formal semantics and a sound and complete resolution-based proof procedure, which makes it also computationally attractive.

9 Space and time-efficient memory layout for multiple inheritance
Peter F. Sweeney, Joseph (Yossi) Gil
October 1999 **ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 34 Issue 10
Full text available: pdf(2.30 MB) Additional Information: full citation, abstract, references, citings, index terms

Traditional implementations of multiple inheritance bring about not only an overhead in terms of run-time but also a significant increase in object space. For example, the number of compiler-generated fields in a certain object can be as large as quadratic in the number of its subobjects. The problem of efficient object layout is compounded by the need to support two different semantics of multiple inheritance: shared, in which a base class inherited along distinct ...

10 The direct cost of virtual function calls in C++
Karel Driesen, Urs Hölzle
October 1996 **ACM SIGPLAN Notices , Proceedings of the 11th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 31 Issue 10
Full text available: pdf(2.03 MB) Additional Information: full citation, abstract, references, citings, index terms

We study the direct cost of virtual function calls in C++ programs, assuming the standard implementation using virtual function tables. We measure this overhead experimentally for a number of large benchmark programs, using a combination of executable inspection and processor simulation. Our results show that the C++ programs measured spend a median of 5.2% of their time and 3.7% of their instructions in dispatch code. For "all virtuals" versions of the programs, the median overhead rises to 13. ...

11 Inheritance of interface specifications (extended abstract)
Gary T. Leavens
August 1994 **ACM SIGPLAN Notices , Proceedings of the workshop on Interface definition languages**, Volume 29 Issue 8
Full text available: pdf(607.70 KB) Additional Information: full citation, abstract, citings, index terms

Four alternatives for the semantics of inheritance of specifications are discussed. The information loss and frame axiom problems for inherited specifications are also considered.

12 Combination of inheritance hierarchies
Harold Ossher, William Harrison
October 1992 **ACM SIGPLAN N tices , c nference proceedings n Object-oriented pr gramming systems, languages, and applications**, Volume 27 Issue 10
Full text available: pdf(1.85 MB) Additional Information: full citation, references, citings, index terms

13 A denotational semantics of inheritance and its correctness
W. Cook, J. Palsberg
September 1989 **ACM SIGPLAN Notices , Conference pr ceedings n Object- riented programming systems, languages and applicati ns**, Volume 24 Issue 10
Full text available: pdf(1.10 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents a denotational model of inheritance. The model is based on an intuitive motivation of the purpose of inheritance. The correctness of the model is demonstrated by proving it equivalent to an operational semantics of inheritance based upon the method-lookup algorithm of object-oriented languages. Although it was originally developed to explain inheritance in object-oriented languages, the model shows that inheritance is a general mechanism that may be applied to any form o ...

14 Integrating inheritance and synchronization in Ada9X
Colin Atkinson, David Weller
October 1993 **Proceedings of the conference on TRI-Ada '93**
Full text available: pdf(1.11 MB) Additional Information: full citation, references, citings, index terms

15 Inheritance and constraint-based grammar formalisms
Rémi Zajac
June 1992 **Computational Linguistics**, Volume 18 Issue 2
Full text available: pdf(1.56 MB) Additional Information: full citation, abstract, references, citings
Publisher Site

We describe an approach to unification grammars that integrates two paradigms: the object-oriented approach, which offers multiple inheritance, complex objects with role-value restrictions and role-values equality, querying as subsumption; the relational programming approach, which offers declarativity, logical variables, nondeterminism with backtracking, and existential queries. This approach is embodied in a constraint-based object-oriented formalism. The interpreter of the formalism is descri ...

16 A calculus for overloaded functions with subtyping
Giuseppe Castagna, Giorgio Ghelli, Giuseppe Longo
January 1992 **ACM SIGPLAN Lisp Pointers , Proceedings of the 1992 ACM conference on LISP and functional programming**, Volume V Issue 1
Full text available: pdf(1.08 MB) Additional Information: full citation, abstract, references, citings, index terms

We present a simple extension of typed &lgr;-claculus where functions can be overloaded by adding different "pieces of code". In short, the code of an overloaded function is formed by several branches of code; the branch to execute is chosen, when the function is applied, according to a particular selection rule which depends on the type of the argument. The crucial feature of the present approach is that a subtyping relation is defied among types, such that ...

17 Assessing software libraries by browsing similar classes, functions and relationships
Amir Michail, David Notkin
May 1999 **Proceedings f the 21st international c nference n Software engineering**
Full text available: pdf(1.23 MB) Additional Information: full citation, references, citings, index terms

Keyw rds: assessment, information retrieval, reuse, software libraries

18 Function preconditions in object oriented software
Jean Pierre LeJacq
September 1991 **ACM SIGPLAN Notices**, Volume 26 Issue 10

Full text available: pdf(397.08 KB) Additional Information: full citation, abstract, citings, index terms

Functions in nonobject oriented languages are typically designed to minimize any preconditions which must be satisfied before the function can be legally called. In object oriented languages this prescription can reduce the refinement of the function through derivation. A tradeoff must be made between refinement potential, efficiency, and risk of incorrect usage.

19 Type theories and object-oriented programmimg
Scott Danforth, Chris Tomlinson
March 1988 **ACM Computing Surveys (CSUR)**, Volume 20 Issue 1

Full text available: pdf(4.39 MB) Additional Information: full citation, abstract, references, citings, index terms

Object-oriented programming is becoming a popular approach to the construction of complex software systems. Benefits of object orientation include support for modular design, code sharing, and extensibility. In order to make the most of these advantages, a type theory for objects and their interactions should be developed to aid checking and controlled derivation of programs and to support early binding of code bodies for efficiency. As a step in this direction, this paper surveys a number ...

20 Types and persistence in database programming languages
Malcolm P. Atkinson, O. Peter Buneman
June 1987 **ACM Computing Surveys (CSUR)**, Volume 19 Issue 2

Full text available: pdf(7.91 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Traditionally, the interface between a programming language and a database has either been through a set of relatively low-level subroutine calls, or it has required some form of embedding of one language in another. Recently, the necessity of integrating database and programming language techniques has received some long-overdue recognition. In response, a number of attempts have been made to construct programming languages with completely integrated database management systems. These lang ...

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Terms used **procedural inheritance** Found **40,106** of **147,060**

Sort results by relevance
Display results expanded form
Save results to a Binder
Search Tips
Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next
Best 200 shown Relevance scale

1 Space and time-efficient memory layout for multiple inheritance
Peter F. Sweeney, Joseph (Yossi) Gil
October 1999 **ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 34 Issue 10
Full text available: pdf(2.30 MB) Additional Information: full citation, abstract, references, citings, index terms

Traditional implementations of multiple inheritance bring about not only an overhead in terms of run-time but also a significant increase in object space. For example, the number of compiler-generated fields in a certain object can be as large as quadratic in the number of its subobjects. The problem of efficient object layout is compounded by the need to support two different semantics of multiple inheritance: shared, in which a base class inherited along distinct ...

2 Inheritance concept for signals in object-oriented extensions to VHDL
Guido Schumacher, Wolfgang Nebel
December 1995 **Proceedings of the conference on European design automation**
Full text available: pdf(889.64 KB) Additional Information: full citation, references, citings, index terms

3 Analysis methodology: Optimization via simulation: two-stage NP method with inheritance
Jumi Kim, Sigurdur Ólafsson
December 2002 **Proceedings of the 34th conference on Winter simulation: exploring new frontiers**
Full text available: pdf(140.10 KB) Additional Information: full citation, abstract, references

The nested partitions method is a flexible and effective framework of optimizing large-scale problems with combinatorial structure. In this paper we consider the nested partitions method for simulation optimization and propose a new variant that uses inheritance to speed convergence. The new nested partitions method with inheritance algorithm performs well for when applied to test problems but it also calls for new analysis of convergence.

4 F-logic: a higher-order language for reasoning about objects, inheritance, and scheme

Michael Kifer, Georg Lausen
June 1989 **ACM SIGMOD Rec rd , Pr ceedings f the 1989 ACM SIGMOD internati nal conference on Management of data**, Volume 18 Issue 2

Full text available: pdf(1.58 MB) Additional Information: full citation, abstract, references, citings, index terms

We propose a database logic which accounts in a clean declarative fashion for most of the "object-oriented" features such as object identity, complex objects, inheritance, methods, etc. Furthermore, database schema is part of the object language, which allows the user to browse schema and data using the same declarative formalism. The proposed logic has a formal semantics and a sound and complete resolution-based proof procedure, which makes it also computationally attractive.

5 LPL++: object programming language with built-in inheritance through unification
Ismail H. Toroslu
April 1998 **ACM SIGMIS Database**, Volume 29 Issue 2

Full text available: pdf(1.06 MB) Additional Information: full citation, abstract, index terms

In this paper, an object logic programming language is proposed that captures all of the basic object-oriented concepts in standard logic programming environment. This paper combines and extends two previously proposed models, namely Conery's technique (1988) which uses first-order logic to model objects including all of the basic object-oriented concepts except inheritance, and the LOGIN language of Ait-Kaci and Nasr (1986) which embeds inheritance into unification using typed logic.In this pap ...

Keywords: deductive databases, inheritance, logic programming, object-oriented databases, unification

6 "...And nothing else changes": the frame problem in procedure specifications
Alex Borgida, John Mylopoulos, Raymond Reiter
May 1993 **Proceedings of the 15th international conference on Software Engineering**

Full text available: pdf(1.11 MB) Additional Information: full citation, references, citings

Keywords: formal specification languages, formal specifications, inheritance, proof obligations, semantics of specification languages

7 Behavioral inheritance: concepts, Ada implementation and experience
Stefano Crespi-Reghizzi, Marco De Michele, Stefano Perotta
December 1992 **Proceedings of the conference on TRI-Ada '92**

Full text available: pdf(1.06 MB) Additional Information: full citation, references, citings, index terms

8 An architecture for object management in OIS
Matts Ahlsen, Anders Bjornerstedt, Stefan Britts, Christer Hulten, Lars Soderlund
August 1984 **ACM Transactions on Information Systems (TOIS)**, Volume 2 Issue 3

Full text available: pdf(1.52 MB) Additional Information: full citation, references, citings, index terms

9 Inheritance and constraint-based grammar formalisms
Rémi Zajac
June 1992 **C mputational Linguistics**, Volume 18 Issue 2

Full text available: pdf(1.56 MB) Additional Information: full citation, abstract, references, citings
Publisher Site

We describe an approach to unification grammars that integrates two paradigms: the object-oriented approach, which offers multiple inheritance, complex objects with role-value restrictions and role-values equality, querying as subsumption; the relational programming approach, which offers declarativity, logical variables, nondeterminism with backtracking, and existential queries. This approach is embodied in a constraint-based object-oriented formalism. The interpreter of the formalism is descri ...

10 Omega: a uniform object model easy to gain Ada's ends
Xianzhong Liang, Zhenyu Wang
June 2001 **ACM SIGAda Ada Letters**, Volume XXI Issue 2

Full text available: pdf(1.02 MB) Additional Information: full citation, abstract, references, citings

Ada provides full capacities of supporting object-orientation, but multiplex object patterns introduced by Ada are so complex that it is difficult for novices to master them. Investigating object patterns implemented by package, protected and task units, this paper presents a uniform object model (Omega), so as to simplify object-orientation with Ada. And exploiting Ada95's capacities, an approach of an Omega pre-processor is also made in confirmation of the model, which rationally hides from th ...

11 Understanding object-oriented: a unifying paradigm
Tim Korson, John D. McGregor
September 1990 **Communications of the ACM**, Volume 33 Issue 9

Full text available: pdf(2.30 MB) Additional Information: full citation, abstract, references, citings, index terms, review

The need to develop and maintain large complex software systems in a competitive and dynamic environment has driven interest in new approaches to software design and development. The problems with the classical waterfall model have been cataloged in almost every software engineering text [19,23]. In response, alternative models such as the spiral [2], and fountain [9] have been proposed. Problems with traditional development using the classical life cycle include no iteration, no ...

12 Types and persistence in database programming languages
Malcolm P. Atkinson, O. Peter Buneman
June 1987 **ACM Computing Surveys (CSUR)**, Volume 19 Issue 2

Full text available: pdf(7.91 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Traditionally, the interface between a programming language and a database has either been through a set of relatively low-level subroutine calls, or it has required some form of embedding of one language in another. Recently, the necessity of integrating database and programming language techniques has received some long-overdue recognition. In response, a number of attempts have been made to construct programming languages with completely integrated database management systems. These lang ...

13 Unified dialogue management in the carousel system
Erik Sandewall
January 1982 **Proceedings of the 9th ACM SIGPLAN-SIGACT symposium n Principles of programming languages**

Full text available: pdf(1.23 MB) Additional Information: full citation, abstract, references, citings

The paper describes the design for a software environment which unifies the dialogue management support of a number of existing software tools, such as command-language handlers, operating-system shells, transition diagram interpreters, and forms management

systems. The unification is accomplished without undue complexity: the resulting design is clean and inspectable.A key notion in the design is the use of three orthogonal abstraction hierarchies, for interaction contexts, for interactive oper ...

14 Semantic analysis of virtual classes and tested classes

Ole Lehrmann Madsen

October 1999 **ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 34 Issue 10

Full text available: pdf(1.82 MB) Additional Information: full citation, abstract, references, citings, index terms

Virtual classes and nested classes are distinguishing features of BETA. Nested classes originated from Simula, but until recently they have not been part of main stream object-oriented languages. C++ has a restricted form of nested classes and they were included in Java 1.1. Virtual classes is the BETA mechanism for expressing generic classes and virtual classes is an alternative to parameterized classes. There has recently been an increasing interest in virtual classes and a number of pro ...

Keywords: generic class, parameterized class, semantic analysis, virtual class

15 On understanding types, data abstraction, and polymorphism

Luca Cardelli, Peter Wegner

December 1985 **ACM Computing Surveys (CSUR)**, Volume 17 Issue 4

Full text available: pdf(4.20 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Our objective is to understand the notion of *type* in programming languages, present a model of typed, polymorphic programming languages that reflects recent research in type theory, and examine the relevance of recent research to the design of practical programming languages. Object-oriented languages provide both a framework and a motivation for exploring the interaction among the concepts of type, data abstraction, and polymorphism, since they extend the notion of type t ...

16 Unreachable procedures in object-oriented programming

Amitabh Srivastava

December 1992 **ACM Letters on Programming Languages and Systems (LOPLAS)**, Volume 1 Issue 4

Full text available: pdf(579.19 KB) Additional Information: full citation, abstract, references, citings, index terms

Unreachable procedures are procedures that can never be invoked. Their existence may adversely affect the performance of a program. Unfortunately, their detection requires the entire program to be present. Using a long-time code modification system, we analyze large, linked, program modules of C++, C, and Fortran. We find that C++ programs using object-oriented programming style contain a large fraction of unreachable procedure code. In contrast, C and Fortran programs have a low and essent ...

Keywords: link-time code modification system

17 Managing geometric complexity with enhanced procedural models

Phil Amburn, Eric Grant, Turner Whitted

August 1986 **ACM SIGGRAPH Computer Graphics , Proceedings f the 13th annual conference on C mputer graphics and interactive techniques**, Volume 20 Issue 4

Full text available: pdf(1.70 MB) Additional Information: full citation, abstract, references, citings, index terms

We illustrate two enhancements to procedural geometric models which allow autonomous procedures to jointly satisfy mutual constraints. One of the techniques adds communications paths between procedures which may affect one another. Conflicts are resolved by modifying communicating procedures as they execute.The second technique is a generalization of widely used subdivision procedures. The termination test of typical subdivision methods is replaced with a "transition" test. The subdivision proce ...

18 Type substitution for object-oriented programming

Jens Palsberg, Michael I. Schwartzbach

September 1990 **ACM SIGPLAN Notices , Proceedings of the European conference on object-oriented programming on Object-oriented programming systems, languages, and applications**, Volume 25 Issue 10

Full text available: pdf(989.92 KB) Additional Information: full citation, abstract, references, citings, index terms

Genericity allows the substitution of types in a class. This is usually obtained through parameterized classes, although they are inflexible since any class can be inherited but is not in itself parameterized. We suggest a new genericity mechanism, type substitution, which is a subclassing concept that complements inheritance: any class is generic, can be "instantiated" gradually without planning, and has all of its generic instances as subclasses.

19 An object-oriented framework of pattern recognition systems

Norihiko Yoshida, Kouji Hino

January 1988 **ACM SIGPLAN Notices , Conference proceedings on Object-oriented programming systems, languages and applications**, Volume 23 Issue 11

Full text available: pdf(796.44 KB) Additional Information: full citation, abstract, references, index terms

In this paper, we describe a purely object-oriented framework of pattern recognition systems. Its aim is in dealing with knowledge representation issues in pattern recognition. In our approach, everything works in an entirely autonomous and decentralized manner. Even a search procedure for sample-concept matching is distributed onto every concept object itself by being implemented in what we introduced as the recursive agent-blackboard model. We developed an experimental prototype of charac ...

20 What does Modular-2 need to fully support object oriented programming?

Joseph Bergin, Stuart Greenfield

March 1988 **ACM SIGPLAN Notices**, Volume 23 Issue 3

Full text available: pdf(837.01 KB) Additional Information: full citation, citings, index terms